

Instruction Sets Ch 9-10

Characteristics

- Operands

- Operations

- Addressing

Instruction Formats

15.11.1999

Copyright Teemu Kerola 1999

1

Instruction Set

(käskykanta)

- Collection of instructions that CPU understands
- Only interface to CPU from outside
- CPU executes a program \Leftrightarrow CPU executes given instructions “one at a time”
 - fetch-execute cycle

Fig. 9.1

15.11.1999

Copyright Teemu Kerola 1999

2

Machine Instruction

- Opcode
 - What should I do? Math? Move? Jump?
- Source operand references
 - Where is the data to work on? Reg? Memory?
- Result operand reference
 - Where should I put the result? Reg? Memory?
- Next instruction reference
 - Where is the next instruction? Default? Jump?

Fig. 9.1

15.11.1999

Copyright Teemu Kerola 1999

3

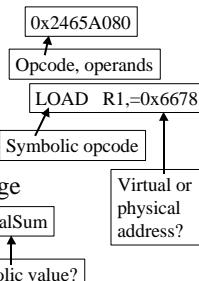
Instruction Representation

- Bit presentation:
 - binary program
- Assembly language
 - symbolic program
- Symbolic assembly language

Fig. 9.11

Copyright Teemu Kerola 1999

4



Instruction Set Design ⁽⁵⁾

- Operation types (operaatiotyppi)
 - How many? What type? Simple? Complex?
- Data types (tietotyppi)
 - Just a few? Many?
- Instruction format (käskyn muoto)
 - fixed length? Varying length? Nr of operands?
- Number of addressable registers
 - too many \Rightarrow long instructions
- Addressing (tiedon osoitus)
 - What modes to use to address data and when?

15.11.1999

Copyright Teemu Kerola 1999

5

Good Instruction Set ⁽²⁾

- Good target to compiler
 - Easy to compile?
 - Easy to compile code that runs fast?
 - Possible to compile code that runs fast?
- Allows fast execution of programs
 - How many meaningless instructions per second?
 - How fast does my program run?
 - Solve linear system of 1000 variables?
 - Set of data base queries?

15.11.1999

Copyright Teemu Kerola 1999

6

Good Instruction Set (contd) (5)

- Beautiful & Aesthetic
 - Orthogonal (ortogonaalinen)
 - Simple, no special registers, no special cases, any data type or addressing mode can be used with any instruction
 - Complete (täydellinen)
 - Lots of operations, good for all applications
 - Regular (säännöllinen)
 - Specific instruction field has always same meaning
 - Streamlined (virtavivainen)
 - Easy to define what resources are used

15.11.1999

Copyright Teemu Kerola 1999

7

Good Instruction Set (contd) (2)

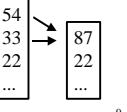
- Easy to implement
 - 18 months vs. 36 months?
 - Who will be 1st in market? Who will get development monies back and who will not?
- Scalability (skaalautuva)
 - Speed up clock speed 10X, does it work?
 - Double address length, does design extend?
 - E.g., 32 bits \Rightarrow 64 bits \Rightarrow 128 bits?

15.11.1999

Copyright Teemu Kerola 1999

8

Number of Operands?

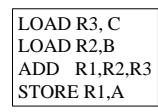
- 3? ADD A,B,C Mem(A) \leftarrow mem(B) + mem(C)
 - Normal case now ADD R1,R2,R3 r1 \leftarrow r2+r3
- 2? ADD R1,R2 r1 \leftarrow r1+r2
 - 1 operand and result the same
- 1? ADD A acc \leftarrow acc+mem(A)
 - 1 operand and result in implicit accumulator
- 0?
 - All operands and ADD result in implicit stack


15.11.1999

Copyright Teemu Kerola 1999

9

Instruction Set Architecture (ISA) Basic Classes

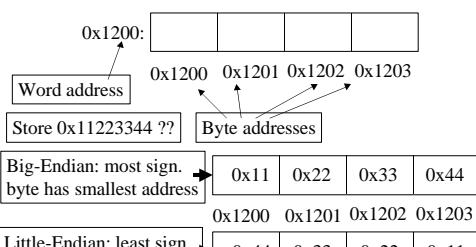
- Accumulator
- Stack
- General Purpose Register
 - only one type of registers, good for all
 - 2 or 3 operands
- Load/Store
 - only load/store instructions access memory
 - 3 operand ALU instructions
 

15.11.1999

Copyright Teemu Kerola 1999

10

Big vs. Little Endian (3)

- How are multi-byte values stored
 

15.11.1999

Copyright Teemu Kerola 1999

11

Big vs. Little Endian

- Address of multi-byte data items is the same in both representations
- Only internal byte order varies
- Must decide one way or the other
 - Math circuits must know which presentation used
 - Must consider when moving data via network
- Power-PC: bi-endian - both modes at use
 - can change it per process basis
 - kernel mode selected separately

15.11.1999

Copyright Teemu Kerola 1999

12

Data (Operands, Result) Location

- Register
 - close, fast

| | |
|----------------|---------|
| acc | r2, r8 |
| register stack | f4, f15 |
 - limited number of them
 - need to load/store values from/to memory sometimes (often)
 - Big problem! 50% of compiler time to decide
 - register allocation problem
- Memory
 - far away

| | |
|---|----------|
| memory stack (hw regs have mem addresses) | 0x345670 |
|---|----------|
 - only possibility for large data sets
 - vectors, arrays, sets, tables, objects, ...

15.11.1999

Copyright Teemu Kerola 1999

13

Aligned Data ⁽⁴⁾

| | |
|--|--------------|
| 2 byte (16-bit) half-word has byte address: | 0010...10010 |
| 4 byte (32-bit) word has byte address: | 0010...10100 |
| 8 byte (64-bit) doubleword has byte address: | 0010...11000 |

- Aligned data
 - faster memory access

| | | | |
|----|----|----|----|
| 11 | 22 | 33 | 44 |
|----|----|----|----|
 - 32-bit data loaded as one memory load
- Non-aligned data
 - saves mem, more bus traffic!

| | | |
|----|----|----|
| | 11 | 22 |
| 33 | 44 | |
 - 32-bit non-aligned data requires 2 memory loads (each 4 bytes) and combining data into one 32-bit data item

15.11.1999

Copyright Teemu Kerola 1999

14

Data Types ⁽⁸⁾

- Address

| |
|----------------------|
| 16b, 32b, 64b, 128b? |
|----------------------|
- Integer

| |
|----------------|
| 16b, 32b, 64b? |
|----------------|
- Floating point

| |
|----------------|
| 32b, 64b, 80b? |
|----------------|
- Decimal

| |
|-------------------------------------|
| 18 digits (9 bytes) packed decimal? |
|-------------------------------------|
- Character

| |
|----------------------------------|
| 1 byte = 8b IRA = ASCII, EBCDIC? |
|----------------------------------|
- String

| |
|---------------------------|
| finite, arbitrary length? |
|---------------------------|
- Logical data

| |
|-----------------------------------|
| 1 bit (Boolean value, bit field)? |
|-----------------------------------|
- Vector, array, record,

15.11.1999

Copyright Teemu Kerola 1999

15

Size of Operand

- 1 word, 32 bits

| |
|------------------|
| int, float, addr |
|------------------|
- 2 words, 64 bits

| |
|--------------------|
| double float, addr |
|--------------------|
- 4 words, 128 bits

| |
|------|
| addr |
|------|
- 1 byte (8 bits)

| |
|------|
| char |
|------|
- 2 bytes

| |
|-----------|
| short int |
|-----------|
- 1 bit

| |
|----------------|
| logical values |
|----------------|

15.11.1999

Copyright Teemu Kerola 1999

16

Pentium II Data Types

- General data types
 - 8-bit byte
 - 16-bit word
 - 32-bit doubleword
 - 64-bit quadword
- Not aligned
- Big Endian
- Specific data types

| |
|-----------|
| Table 9.2 |
|-----------|
- Numerical data types

| |
|------------|
| Figure 9.4 |
|------------|

15.11.1999

Copyright Teemu Kerola 1999

17

Table 9.3

Operation Types

- Data transfer
 - CPU ↔ memory
- ALU operations
 - INT, FLOAT, BOOLEAN, SHIFT, CONVERSION
- I/O
 - read from device, start I/O operation
- Transfer of control
 - jump, branch, call, return, IRET, NOP
- System control
 - HALT, SYSENTER, SYSEXIT, ...

| |
|-----------|
| Table 9.4 |
|-----------|
 - CPUID returns current HW configuration
 - size of L1 & L2 caches, etc

15.11.1999

Copyright Teemu Kerola 1999

18

Data References

- Where is data?
 - in memory
 - in registers
 - in instruction itself
- How to refer to data?
 - various addressing modes
 - multi-phase data access
 - how is data location determined (addressing mode)
 - compute data address (register? effective address?)
 - access data

15.11.1999

Copyright Teemu Kerola 1999

19

Copyright Teemu Kerola 1999

20

Addressing Modes (Ch 10)

Fig. 10.1

- Immediate Data in instruction Table. 10.1
- Direct Memory address of data in instruction
- Indirect Address of memory address of data in instruction (pointer)
- Register Data in register (best case?)
- Register Indirect Register has memory address (pointer)
- Displacement Addr = reg value + constant
- Stack Data is stack pointed by some register

15.11.1999

Copyright Teemu Kerola 1999

21

Displacement Address

- Effective address = $(R1) + A$
 - Contents of R1
 - Constant from instruction
- Constant is often small (8 bits, 16 bits?)
- Many uses
 - PC relative
 - Base register address
 - Array index
 - Record field
 - Stack references

Copyright Teemu Kerola 1999

22

More Addressing Modes

- Autoincrement $EA = (R), R \leftarrow (R) + S$
 - E.g., R pointer to an array
- Autodecrement $R \leftarrow (R) - S, EA = (R)$
 - E.g., R pointer to an array
- Autoincrement deferred
 - $EA = \text{Mem}(R), R \leftarrow (R) + S$
 - E.g., R pointer to an array of pointers
- Scaled $EA = A + (R_i) + (R_j) * S$
 - E.g., item (R_i, R_j) in 2-dimensional array $A[i,j]$

15.11.1999

Copyright Teemu Kerola 1999

23

Pentium II Addressing Modes

- Immediate
 - 1, 2, 4 bytes
- Register operand
 - 1, 2, 4, 8 byte registers
 - not all registers with every instruction
- Operands in Memory
 - Fig. 10.2
 - compute effective address and combine with segment register to get linear address (virtual address)

15.11.1999

Copyright Teemu Kerola 1999

24

Instruction Format ⁽⁴⁾

- How to represent instructions in memory?
- How long instruction
 - Descriptive or dense? Code size?
- Fast to load?
 - In many parts?
 - One operand description at a time?
- Fast to parse?
 - All instruction same size & same format?
 - Very few formats?

15.11.1999

Copyright Teemu Kerola 1999

25

Instruction Format (contd) ⁽³⁾

- How many addressing modes?
 - Fewer is better, but harder to compile to
- How many operands?
 - 3 gives you more flexibility, but takes more space
- How many registers?
 - 16 regs → need 4 bits to name it
 - 256 regs → need 8 bits to name it
 - need at least 16-32 for easy register allocation

15.11.1999

Copyright Teemu Kerola 1999

26

Instruction Format (contd) ⁽³⁾

- How many register sets?
 - A way to use more registers without forcing long instructions for naming them
 - One register set for each subroutine call?
 - One for indexing, one for data?
- Address range, number of bits in displacement
 - more is better, but it takes space
- Address granularity
 - byte is better, but word address is shorter

15.11.1999

Copyright Teemu Kerola 1999

27

Pentium II Instruction Set ⁽⁵⁾

- CISC - Complex Instruction Set Computer
- At most one memory address
- “Everything” is optional
- “Nothing” is fixed
- Difficult to parse
 - all latter fields and their interpretation depends on earlier fields

Fig. 10.8

15.11.1999

Copyright Teemu Kerola 1999

28

Pentium II Instruction Prefix Bytes ⁽⁴⁾

- Instruction prefix (optional)
 - LOCK - exclusive use of shared memory
 - REP - repeat instruction for string characters
- Segment override (optional)
 - override default segment register
 - default is implicit, no need to store it every instruction
- Address size (optional)
 - use the other (16 or 32 bit) address size
- Operand size (optional)
 - use the other (16 or 32 bit) operand size

Fig. 10.8 (a)

15.11.1999

Copyright Teemu Kerola 1999

29

Pentium II Instruction Fields ⁽³⁾

- Opcode
 - specific bit for byte size data
- Mod r/m (optional)
 - data in reg (8) or in mem?
 - which addressing mode of 24?
 - can also specify opcode further for some opcodes
- SIB (optional)
 - extra field needed for some addressing modes
 - scale for scaled indexing
 - index register
 - base register

Fig. 10.8 (b)

15.11.1999

Copyright Teemu Kerola 1999

30

Pentium II Instruction Fields (contd) (2)

- Displacement (optional)
 - for certain addressing modes
 - 1, 2, or 4 bytes
- Immediate (optional)
 - for certain addressing modes
 - 1, 2, or 4 bytes

Fig. 10.8 (b)

15.11.1999

Copyright Teemu Kerola 1999

31

PowerPC Instruction Format (7)

- RISC - Reduced Instruction Set Computer
- Fixed length, just a few formats
- Only load/store instructions access memory
- Only 2 addressing modes for data
- 32 general purpose registers can be used everywhere
- Fixed data size
 - no string ops
- Simple branches
 - CR-field determines which register to compare
 - L-bit determines whether a subroutine call
 - A-bit determines if branch is absolute or PC-relative

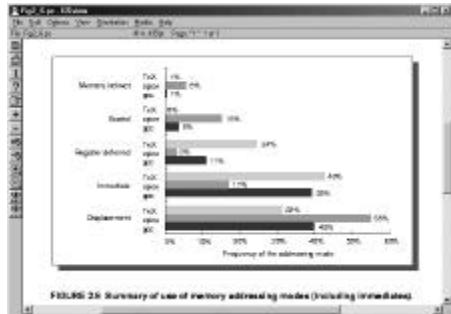
Fig. 10.9

15.11.1999

Copyright Teemu Kerola 1999

32

-- End of Chapters 9-10: Instruction Sets --



(Hennnsey-Patterson, Computer Architecture, 2nd Ed, 1996)

15.11.1999

Copyright Teemu Kerola 1999

33